COMP 430
Intro. to Database Systems
Quantification
Review of SQL conditions

• Compare elements of same record
  • Combine with join to work with multiple records

• Compare aggregations

• Check membership – IN
Existential quantification

\{c \mid \text{Customer}(c) \land \\
(\exists \, o. \text{Order}(o) \land \\
c.\text{id} = o.\text{customer_id})\}\}

SELECT *
FROM Customer
WHERE EXISTS (SELECT *
FROM Order
WHERE Customer.id = customer_id);

EXISTS (subquery) returns a Boolean.
Subquery’s SELECT irrelevant.

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joe</td>
</tr>
<tr>
<td>4</td>
<td>Mary</td>
</tr>
<tr>
<td>3</td>
<td>Scott</td>
</tr>
<tr>
<td>6</td>
<td>Elizabeth</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>customer_id</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>105</td>
<td>4</td>
<td>Shoes</td>
</tr>
<tr>
<td>107</td>
<td>4</td>
<td>Pants</td>
</tr>
<tr>
<td>108</td>
<td>1</td>
<td>Pants</td>
</tr>
<tr>
<td>109</td>
<td>3</td>
<td>Tie</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joe</td>
</tr>
<tr>
<td>4</td>
<td>Mary</td>
</tr>
<tr>
<td>3</td>
<td>Scott</td>
</tr>
</tbody>
</table>
Existential quantification negated

\{c \mid \text{Customer}(c) \land \\
(\neg \exists \; o. \text{Order}(o) \land \\
c.\text{id} = o.\text{customer\_id})\}

SELECT *
FROM Customer
WHERE NOT EXISTS (SELECT *
FROM Order
WHERE Customer.id = customer\_id);

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joe</td>
</tr>
<tr>
<td>4</td>
<td>Mary</td>
</tr>
<tr>
<td>3</td>
<td>Scott</td>
</tr>
<tr>
<td>6</td>
<td>Elizabeth</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>customer_id</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>105</td>
<td>4</td>
<td>Shoes</td>
</tr>
<tr>
<td>107</td>
<td>4</td>
<td>Pants</td>
</tr>
<tr>
<td>108</td>
<td>1</td>
<td>Pants</td>
</tr>
<tr>
<td>109</td>
<td>3</td>
<td>Tie</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>Elizabeth</td>
</tr>
</tbody>
</table>
Existential quantification – efficiency

\{c \mid \text{Customer}(c) \land (\exists o. \text{Order}(o) \land c.\text{id} = o.\text{customer}_\text{id})\}

**SELECT** *
FROM Customer
WHERE EXISTS (SELECT *
FROM Order
WHERE Customer.\text{id} = \text{customer}_\text{id});

Executes subquery for each Customer record.

However, can stop subquery at first satisfying Order.
Existential quantification negated

\{c | \text{Customer}(c) \land \\
\neg \exists \ o. \text{Order}(o) \land \\
c.\text{id} = o.\text{customer_id})\}

\begin{Verbatim}
SELECT *
FROM Customer
WHERE NOT EXISTS (SELECT *
FROM Order
WHERE Customer.id = customer_id);
\end{Verbatim}

Executes subquery for each Customer record.

However, can stop subquery at first non-satisfying Order.
Logical equivalence of quantifiers

\[ \neg \exists x . R(x) = \forall x . \neg R(x) \]

\[ \neg \forall x . R(x) = \exists x . \neg R(x) \]
Universal quantification

• No directly comparable FORALL!
• Generally less useful idea in SQL.
  • E.g., find Customer that has placed all of the Orders.
  • However, see today’s activity.

• $\forall x . R(x) = \neg \exists x. \neg R(x)$
EXISTS vs. other approaches

SELECT *
FROM   Customer
WHERE EXISTS (SELECT *
               FROM   Order
               WHERE Customer.id = customer_id);

SELECT *
FROM   Customer
WHERE id IN (SELECT customer_id
             FROM   Order);

SELECT DISTINCT Customer.id, name
FROM   Customer, Order
WHERE Customer.id = customer_id;

SELECT *
FROM   Customer
WHERE (SELECT Count(*)
       FROM   Order
       WHERE Customer.id = customer_id);
EXISTS vs. INTERSECT

SELECT id
FROM   Customer
WHERE EXISTS (SELECT *
               FROM Order
               WHERE Customer.id = customer_id);

SELECT id
FROM   Customer
INTERSECT
SELECT customer_id
FROM   Order;

Inconvenient if you want more attributes.
Activity – Use EXISTS in queries

Given:
Store (id, name, type)
City (id, name)
CityStore (store_id, city_id)

Warm up:
• The stores that are in at least one city each are of what types?
• The stores that are in no city each are of what types?
• The stores that are present in all cities are of what types?
Another form of existential quantification

```sql
SELECT *  
FROM Customer  
WHERE EXISTS (SELECT *  
              FROM Order  
              WHERE Customer.id = customer_id);
```

```sql
SELECT *  
FROM Customer  
WHERE id = ANY (SELECT customer_id  
                FROM Order);
```

Syntactically strange, but think of ANY as part of the comparison operator.

Can use any comparison operator.

Or SOME.
Another form of universal quantification

```
SELECT *
FROM   Product p1
WHERE p1.price >= ALL (SELECT p2.price
                        FROM   Product p2);
```

```
SELECT *
FROM   Product p1
WHERE NOT (p1.price < ANY (SELECT p2.price
                            FROM   Product p2));
```